TP4 Data Mining

Deniz Sungurtekin

May 2020

1 Introduction

Le but de ce travail pratique est d'approfondir la théorie reçu en cours au sujet de la fonction de logistique binaire notamment sur la façon d'aborder son optimisation en y intégrant la régularisation L2. Après avoir bien compris cette thématique, le travail consistera d'implémenter un classifieur en utilisant la méthode de la descente de gradient et celle de Newton sur un ensemble de donné CIFAR10. Dans un premier temps nous allons nous pencher sur la fonction de coût de la régression logistique et développer sa dérivation au premier puis au second degré. Ensuite nous y ajouterons la régularization L2 et observerons les changements provoqués sur le gradient et la hessienne. Finalement, nous utiliserons ces résultats pour appliquer la méthode de la descente de gradient et celle de Newton afin de construire notre classifieur pour évaluer, analyser et comprendre les résultats obtenues.

2 Développement

1. Avant toute chose assurons nous de posséder les bons outils afin de comprendre la méthodologie utilisée. Soit la fonction de coût de la fonction de régression logistique donné par le négative log de la "likelihood" d'un vecteur de classe $\mathbf{y}$ observé suivant une loi de Bernoulli donné par:

$$E(w) = -ln(p(y|X)) = -ln(\prod_{i=1}^{N} p(C_1|x_i)^{y_i}(1 - p(C_1|x_i))^{1-y_i})$$

En continuant le développement on obtient ainsi notre fonction de coût:

$$-\sum_{i=1}^{N} y_i \, ln(\sigma(w^t x_i)) - \sum_{i=1}^{N}(1 - y_i)ln(1 - \sigma(w^t x_i))$$

$$ou \quad \sigma(x) = \frac{1}{1 + exp(-x)} \quad et \quad w^t x = a(x) = ln(\frac{p(C_1(x))}{p(C_2(x))} \quad avec \quad \sigma(-a) = 1 - \sigma(a).$$

2. On désire maintenant minimiser le coût donc on calcule le gradient de la façon suivante:

   On commence par multiplie par -1 la formule obtenue en 1) afin de simplifier le calcul:

$$-E(w) = \sum_{i=1}^{N} y_i \, ln(\sigma(w^t x_i)) + \sum_{i=1}^{N} (1 - y_i) ln(1 - \sigma(w^t x_i))$$

   On calcule ensuite la derive pour un seul point de donne (x,y), on obtient:

$$\frac{-\delta E(x)}{\delta w_j} = \frac{\delta}{\delta w_j} y \, ln(\sigma(w^t x)) + \frac{\delta}{\delta w_j}(1-y) \, ln(1-\sigma(w^t x)) = [\frac{y}{\sigma(w^t x)} - \frac{1-y}{1-\sigma(w^t x)}] \frac{\delta}{\delta w_j} \sigma(w^t x)$$

$$= [\frac{y}{\sigma(w^t x)} - \frac{1-y}{1-\sigma(w^t x)}] \sigma(w^t x)[1-\sigma(w^t x)] x_i = [\frac{y - \sigma(w^t x)}{\sigma(w^t x)[1-\sigma(w^t x)]}] \sigma(w^t x)[1-\sigma(w^t x)] x_i$$

$$= [y - \sigma(w^t x)] x_i$$ On multiplie par -1 à nouveau afin d obtenir:

$$(\sigma(w^t x) - y_i) x_i$$

   On a donc pour l ensemble des points le resultat suivant:

$$\Delta E(x) = \sum_{i=1}^{N} (\sigma(w^t x) - y_i) x_i = X^t (\sigma - y) \quad où \quad \sigma = (\sigma(w^t x_1), ..\sigma(w^t x_n))^T$$

3. Calculons maintenant la hessienne:

   Notons $\Delta E_i(w)$ le gradient pour un seul element, on a alors:

$$\Delta^2 E_i(w) = \frac{\delta E_i(w)}{\delta w \delta w_j} = x_i x_i^t \sigma(w^t x_i)(1 - \sigma(1 - \sigma(w^t x_i))$$

   pour n element on obtient alors: $\sum_{i=1}^{N} x_i x_i^t \sigma(w^t x_i)(1 - \sigma(w^t x_i))$

   où $\sum_{i=1}^{N} x_i x_i^t = X X^t$ et $\sigma(w^t x_i)(1 - \sigma(w^t x_i)) = R_{i,i}$

   donc la hessienne est donné par: $\Delta^2 E(w) = XRX^t = X^t RX <=> R = diag(\sigma_i(1-\sigma_i)) \, , i = 1..N$

4. Voici désormais la fonction de coût en ajoutant un régularisateur L2:

   $-\sum_{i=1}^{N} y_i ln(\sigma(w^t x_i)) - \sum_{i=1}^{N} (1-y_i) ln(1-\sigma(w^t x_i)) + \frac{\lambda}{2} \sum_{j=1}^{d} w_j^2$ où d est le nombre de features

   Remarque: On divise lambda par deux afin de simplifier le calcul des dérivées

5. Calculons à nouveau le gradient:

   Puisqu'on a uniquement ajouter un terme, il suffit d'ajouter: $\frac{\delta}{\delta w_j}\frac{\lambda}{2}||w||^2 = \lambda w_j$

   donc: $\Delta E(x) = \sum_{i=1}^{N}[(\sigma(w^t x) - y_i)x_i] + \lambda w = X^t(\sigma - y) + \lambda w$

6. Calculons à nouveau la hessienne:

   Le raissonement est le même, il suiffit d'ajouter: $\frac{\delta}{\delta w_j}\lambda w_j = \lambda$

   donc on obtient: $\Delta^2 E(w) = \sum_{i=1}^{N} x_i x_i^t \sigma(w^t x_i)(1 - \sigma(w^t x_i)) + \lambda$

Avec tout ces éléments en mains nous pouvons désormais procéder à l'implémentation de la régression logistique sur notre ensemble de donné.

3 Implementation et Résultats

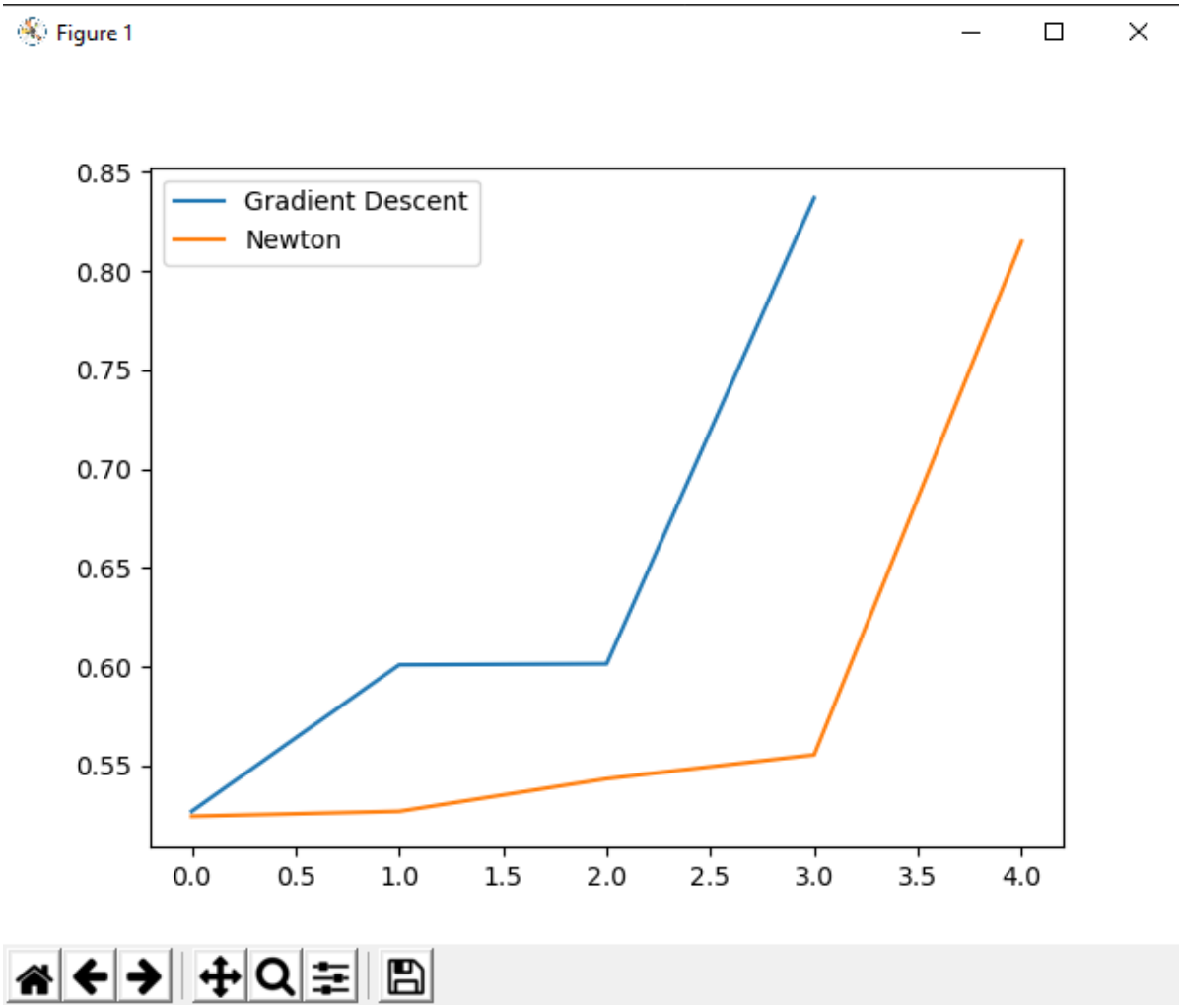
Comme demandé, j'ai donc complété toute les fonctions nécessaires dans le fichier logistic-regression.py, comme vous pouvez le constater toute mes fonctions passent les tests mis à par la fonction "predict" qui me retourne une erreur de 255 lorsqu'elle compare ma somme sur les éléments du vecteur y-pred à la votre. Malgré une vérification, je n'ai pas pu trouver l'origine de cette erreur, toutefois j'ai estimé que cette erreur est petite sur un ensemble de donné de 9000 éléments et donc décidé de continuer ainsi.

En ce qui concerne l'implémentation de la méthode train() dans le classifier.py, je n'ai rencontré aucun problème. J'ai donc simplement appliqué la formule de Newton en utilisant le gradient et la hessienne et la descente de Gradient donné en cours.

Voici donc les résultats obtenus pour 10 itérations de Newton et 300 de la descente en gradient:

```
best validation GD: 0.837
best validation NW: 0.815
```

et voici le graphique montrant l'impact des nouveaux poids w calculé après une itération:



On observe une efficacité presque équivalente alors qu'on a 30 fois moins d'itérations pour Newton.

En ce qui concerne l'impact du "learning rate", on voit que cet hyperparametre contrôle la manière dont on va changer le modèle en réponse à une erreur estimé à chaque fois qu'on met à jour les poids w. Pour la "regularization strenghts", on remarque qu'il n'améliore pas forcément la vitesse de la convergence des poids vers ses valeurs optimale mais permet d'améliorer la performance globale de l'algorithme. On peut la comparer à l'ajout d'un bias à notre model si il possède une trop grande variance, afin d'empècher un "overfitting" des donnés d'entraînements, cependant si le regularizateur est trop grand on observera un "underfitting".

4 Conclusion

Finalement, on retient que l'efficacité de la régression logistique dépend essentiellement de l'estimation des poids w qui dépendent du "learning rate" et du coefficient de regularization. En résumé, on a vu qu'il était nécessaire de calculer le gradient de la "loss function" afin d'utiliser la méthode de la descente du gradient afin d'estimer nos coefficients. Pour améliorer la précision de cette estimation, il est donc aussi possible d'utiliser la méthode de Newton en calculant également la hessienne, cependant cette dernière est bien plus coûteuse en temps.